

*eunix: whoami*

*Eric Bailey*

*November 29, 2017*<sup>1</sup>

<sup>1</sup> Last updated July 29, 2020

A reimplementaion of `whoami` for my own edification.

**1a** `< * 1a>≡`  
`<Include headers. 2a>`

## *Contents*

*The main Function* 1

*Include Headers* 2

*The usage Function* 3

*Processing Options* 3

*Printing the Current User's Name* 3

*Full Listing* 5

*Chunks* 6

*Index* 6

`<Define constants. 3d>`

`<Forward declarations. 3a>`

`<Define the main function. 1b>`

`<Define the usage function. 3b>`

Root chunk (not used in this document).

## *The main Function*

**1b** `<Define the main function. 1b>≡`  
`int main(int argc, char *argv[])`  
`{`

`<Process given options. 3c>`

`<Print the user name associated with the current effective user ID. 3e>`

`return 0;`

`}`

This code is used in chunk **1a**.

Defines:

`argc`, used in chunk **3c**.

`argv`, used in chunk **3c**.

`main`, never used.

## Include Headers

Include the core input and output functions from the C standard library.

2a `<Include headers. 2a>≡`  
`#include <stdio.h>`

This definition is continued in chunk 2.

This code is used in chunk 1a.

Defines:

`EOF`, used in chunk 3c.

`printf`, used in chunk 4d.

From `sys/types.h` import `uid_t`, a data type for user IDs.

2b `<Include headers. 2a>+≡`  
`#include <sys/types.h>`

This code is used in chunk 1a.

Defines:

`uid_t`, used in chunk 3.

From `pwd.h` import the struct, `passwd`, which notably includes the member, `pw_name`, and has a constructor function, `getpwuid`.

2c `<Include headers. 2a>+≡`  
`#include <pwd.h>`

This code is used in chunk 1a.

Defines:

`getpwuid`, used in chunk 4c.

`passwd`, used in chunk 3e.

`pw->pw_name`, used in chunk 4e.

From `unistd.h` import the function, `geteuid`, which returns the effective user ID of the calling process.

2d `<Include headers. 2a>+≡`  
`#include <unistd.h>`

This code is used in chunk 1a.

Defines:

`geteuid`, used in chunk 4a.

Include the GNU `getopt` function from the GNU C Library.

2e `<Include headers. 2a>+≡`  
`#include <getopt.h>`

This code is used in chunk 1a.

Defines:

`getopt`, used in chunk 3c.

“The `getopt` function gets the next option argument from the argument list specified by the `argv` and `argc` arguments. Normally these values come directly from the arguments received by `main`.” – GNU, 2017

### The *usage* Function

Define the **usage** function, which displays information about how to use **whoami**.

**3a** *<Forward declarations. 3a>≡*  
**void usage();**  
 This code is used in chunk **1a**.  
 Uses **usage 3b**.

**3b** *<Define the usage function. 3b>≡*  
**void usage()**  
**{**  
     **fprintf(stderr, "Try 'whoami -help' for more information.\n");**  
**}**

This code is used in chunk **1a**.

Defines:

**usage**, used in chunk **3**.

### Processing Options

If any options are given, complain about the first one (via **getopt**), print the **usage** information, and return a nonzero status code.

**3c** *<Process given options. 3c>≡*  
**if (getopt(argc, argv, "") ≠ EOF) {**  
     **usage();**  
     **return 1;**  
**}**

This code is used in chunk **1b**.

Uses **argc 1b**, **argv 1b**, **EOF 2a**, **getopt 2e**, and **usage 3b**.

### Printing the Current User's Name

Define a constant, **NO\_UID**, to represent the case when **geteuid** returns -1, which in **whoami** will signify failure to find the user ID.

**3d** *<Define constants. 3d>≡*  
**uid\_t NO\_UID = -1;**

This code is used in chunk **1a**.

Defines:

**NO\_UID**, used in chunk **4b**.

Uses **uid\_t 2b**.

Declare the variables **uid**, to store the current user ID, and **pw**, to store further information about the current user.

**3e** *<Print the user name associated with the current effective user ID. 3e>≡*  
**uid\_t uid;**  
**struct passwd \*pw;**

This definition is continued in chunk **4**.

This code is used in chunk **1b**.

Defines:

**pw**, used in chunk **4**.

**uid**, used in chunk **4**.

Uses **passwd 2c** and **uid\_t 2b**.

Get the effective user ID and store it as `uid`.

4a *<Print the user name associated with the current effective user ID. 3e>+≡*  
`uid = geteuid();`

This code is used in chunk 1b.  
 Uses `geteuid` 2d and `uid` 3e.

Check whether the effective user ID is `NO_UID`, in which case we won't be able to *<find a user with a matching uid 4c>*.

4b *<the user ID is NO\_UID 4b>≡*  
`uid = NO_UID`

This code is used in chunk 4d.  
 Uses `NO_UID` 3d and `uid` 3e.

Search the user database for an entry with a matching `uid`. If `getpwuid` fails, it returns a null pointer.

4c *<find a user with a matching uid 4c>≡*  
`pw = getpwuid(uid)`

This code is used in chunk 4d.  
 Uses `getpwuid` 2c, `pw` 3e, and `uid` 3e.

If *<the user ID is NO\_UID 4b>* or we're unable to *<find a user with a matching uid 4c>*, print a descriptive error message and return a nonzero status code.

4d *<Print the user name associated with the current effective user ID. 3e>+≡*  
`if ((<the user ID is NO_UID 4b> || !(<find a user with a matching uid 4c>))) {`  
 `printf("Cannot find name for user ID %lu\n",`  
 `(unsigned long int) uid);`  
 `return 1;`  
`}`

This code is used in chunk 1b.  
 Uses `printf` 2a and `uid` 3e.

4e *<Print the user name associated with the current effective user ID. 3e>+≡*  
`puts(pw->pw_name);`

This code is used in chunk 1b.  
 Uses `pw` 3e and `pw->pw_name` 2c.

*Full Listing*

```

1  #include <stdio.h>
2  #include <sys/types.h>
3  #include <pwd.h>
4  #include <unistd.h>
5  #include <getopt.h>
6
7
8  uid_t NO_UID = -1;
9
10
11 void usage();
12
13
14 int main(int argc, char *argv[])
15 {
16     if (getopt(argc, argv, "") != EOF) {
17         usage();
18         return 1;
19     }
20
21     uid_t uid;
22     struct passwd *pw;
23     uid = geteuid();
24
25     if (uid == NO_UID || !(pw = getpwuid(uid))) {
26         printf("Cannot find name for user ID %lu\n",
27             (unsigned long int) uid);
28         return 1;
29     }
30     puts(pw->pw_name);
31
32     return 0;
33 }
34
35
36 void usage()
37 {
38     fprintf(stderr, "Try 'whoami --help' for more information.\n");
39 }

```

*Chunks*

⟨\* [1a](#)⟩ [1a](#)  
 ⟨Define constants. [3d](#)⟩ [1a](#), [3d](#)  
 ⟨Define the **main** function. [1b](#)⟩ [1a](#), [1b](#)  
 ⟨Define the **usage** function. [3b](#)⟩ [1a](#), [3b](#)  
 ⟨find a user with a matching **uid** [4c](#)⟩ [4c](#), [4d](#)  
 ⟨Forward declarations. [3a](#)⟩ [1a](#), [3a](#)  
 ⟨Include headers. [2a](#)⟩ [1a](#), [2a](#), [2b](#), [2c](#), [2d](#), [2e](#)  
 ⟨Print the user name associated with the current effective user  
     ID. [3e](#)⟩ [1b](#), [3e](#), [4a](#), [4d](#), [4e](#)  
 ⟨Process given options. [3c](#)⟩ [1b](#), [3c](#)  
 ⟨the user ID is **NO\_UID** [4b](#)⟩ [4b](#), [4d](#)

*Index*

argc: [1b](#), [3c](#)  
 argv: [1b](#), [3c](#)  
 EOF: [2a](#), [3c](#)  
 geteuid: [2d](#), [4a](#)  
 getopt: [2e](#), [3c](#)  
 getpwuid: [2c](#), [4c](#)  
 main: [1b](#)  
 NO\_UID: [3d](#), [4b](#)  
 passwd: [2c](#), [3e](#)  
 printf: [2a](#), [4d](#)  
 pw: [3e](#), [4c](#), [4e](#)  
 pw→pw\_name: [2c](#), [4e](#)  
 uid: [3e](#), [4a](#), [4b](#), [4c](#), [4d](#)  
 uid\_t: [2b](#), [3d](#), [3e](#)  
 usage: [3a](#), [3b](#), [3c](#)

*References*

GNU. The GNU C Library: Using the getopt function. [https://www.gnu.org/software/libc/manual/html\\_node/Using-Getopt.html](https://www.gnu.org/software/libc/manual/html_node/Using-Getopt.html),  
 2017. Accessed: 2017-11-05.